Navigating in Complex Business Processes*

Markus Hipp[1], Bela Mutschler[2], and Manfred Reichert[3]

[1] Group Research & Advanced Engineering, Daimler AG, Germany
markus.hipp@daimler.com
[2] University of Applied Sciences Ravensburg-Weingarten, Germany
bela.mutschler@hs-weingarten.de
[3] Institute of Databases and Information Systems, University of Ulm, Germany
manfred.reichert@uni-ulm.de

Abstract. In order to provide information needed in knowledge-intense business processes, large companies often establish intranet portals, which enable access to their process handbook. Especially, for large business processes comprising hundreds or thousands of process steps, these portals can help to avoid time-consuming access to paper-based process documentation. However, business processes are usually presented in a rather static manner within these portals, e.g., as simple drawings or textual descriptions. Companies therefore require new ways of making large processes and process-related information better explorable for end-users. This paper picks up this issue and presents a formal navigation framework based on linear algebra for navigating in large business processes.

Key words: Process Navigation, Process Visualization

1 Introduction

Large, knowledge-intense business processes, like the ones for engineering the electric-/electronic components in a car [1], may comprise hundreds or thousands of process steps. Usually, each process step is associated with task-related information, like engineering documents, development guidelines, contact information, or tool instructions—denoted as *process information*. To handle such a large information space (cf. Fig. 1), companies use web-based intranet portals. The goal is to provide a central point of access for their staff members enabling them to quicker find and access the process information needed. Process information, however, are often manually linked within these portals and hard-wired navigation structures are used to explore them. Process information not linked at all is not directly accessible for users.

In these portals, business processes and process information are usually visualized in a rather static manner [2, 3], e.g., in terms of simple document lists (cf. Fig. 1). Van Wijk has shown that such visualizations often result in an information overload, rather disturbing than supporting the user [4]. Furthermore,

* This research has been done in the niPRO project funded by the German Federal Ministry of Education and Research (BMBF) under grant number 17102X10.

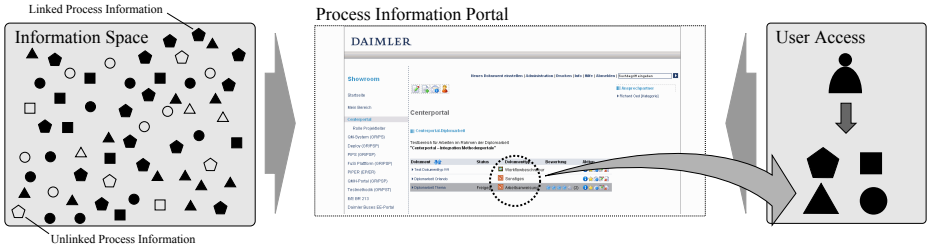

Fig. 1: Providing process information in intranet portals.

process participants may have different perspectives on a business process and related process information [5, 6]. For example, consider the development of an ABS[2] control unit:

- **Requirement Engineers (Use Case 1):** Requirement engineers write a general specification for the ABS control unit. For this purpose, they need detailed instructions, templates, and contact persons. Needed information are also logic relations between process steps and process information.
- **Project Managers (Use Case 2):** Project managers must be able to identify the reasons for missed project deadlines, which negatively affect overall project goals. In this context, they need information about the status of all process steps as well as an abstract view on process steps and associated process information (e.g. due dates and duration of process steps).

These two use cases illustrate the diversity of process tasks and related process information needed. Obviously, users may have different roles and hence follow different navigation goals. Requirement engineers, for example, need very detailed information, whereas project managers ask for information on a more abstract level.

This paper introduces a process navigation framework that allows different users to intuitively and effectively navigate in and explore complex business process models. Our approach provides both processes and process information on different levels of abstraction. In particular, users can dynamically reach their navigation goal independent of their specific role. To provide a sound foundation of this navigation framework, linear algebra is used. We further demonstrate the applicability of the framework along scenarios from the automotive domain.

The remainder of this paper is organized as follows: Section 2 introduces basic notations and our core navigation model. Section 3 describes advanced concepts of our process navigation framework in detail. Section 4 applies these concepts to a real-world scenario. Section 5 discusses related work. Finally, Section 6 concludes the paper with a summary and an outlook.

[2] Antilock Braking System

2 Basic Notations and Core Navigation Model

We first introduce basic notation needed for the understanding of our navigation framework. Specifically, we reuse an existing navigation concept for complex information spaces to business processes—Google Earth [7].

Navigation Dimensions. We distinguish between geographic, semantic, and view navigation dimension. The *geographic dimension* allows for visual zooming without changing the level of detail. Regarding a process model with three process steps, for example, the user may zoom into the first step labeled "General Specification" (cf. Fig. 2a). A metaphor reflecting this dimension is using a magnifier, while reading a newspaper. In the *semantic dimension*, process information may be displayed at different levels of detail. Assuming that process steps comprise multiple activities, these activities may be additionally displayed by increasing the value of the semantic dimension (cf. Fig. 2b). Finally, the *view dimension* allows users to emphasize specific information, while reducing other, e.g., the duration of process steps. For example, the view may change from a logic-based to a time-based one (cf. Fig. 2c). Overall, these three dimensions define the *navigation space*.

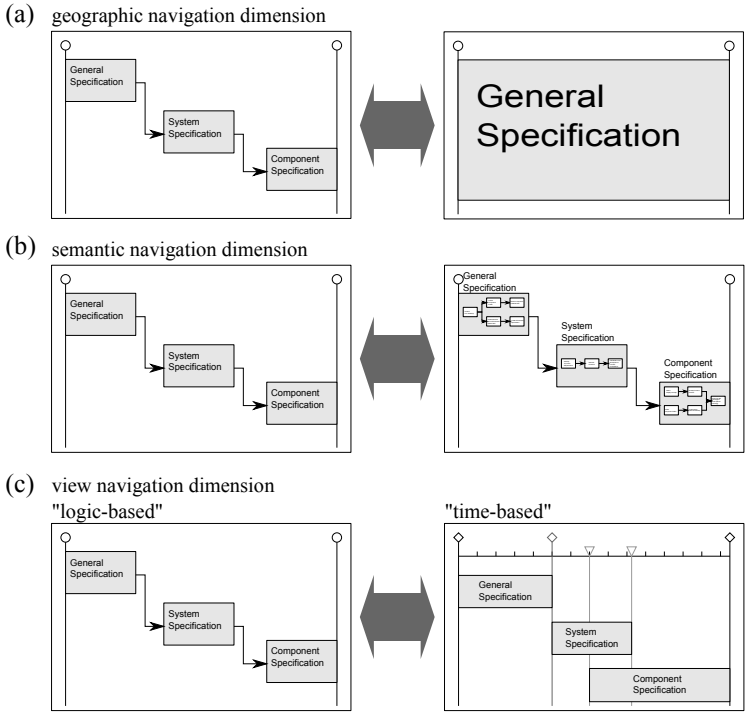

Fig. 2: Different navigation dimensions.

Navigation State (NS). A navigation state corresponds to a specific point within the navigation space. The three navigation dimensions of this space are scaled

in different values of which each represents a relative scale. For the sake of simplicity, we use natural numbers for this. Hence, in our context, we can define a navigation state as a triple. Let g be the value of the geographic dimension, s the one of the semantic dimension, and v the value of the view dimension. A specific navigation state NS can then be represented as:

$$NS = (g, s, v) \text{ with } g, s, v \in \mathbb{N} \tag{1}$$

Note that g, s, and v may be manually selected by the user. The set of all possible navigation states NS_{total} is as follows:

$$NS_{total} = \{(g, s, v) | g, s, v \in \mathbb{N}\} \tag{2}$$

Some of these navigation states may not make sense from a semantical point of view, i.e., they disturb the user or are forbidden by definition. Think of Google Earth and assume the user wants to see the whole globe (geographic dimension) and all city names at the same time (semantic dimension). In such a navigation state, labels would significantly overlap due to limited screen space. Hence, such a navigation state should be not reachable and be added to the set of forbidden navigation states $NS_{forbidden}$. In turn, we denote the set of allowed navigation states as *basis model* BM.

Basis Model (BM). The basis model corresponds to the set of allowed navigation states within the navigation space:

$$BM = NS_{total} \backslash NS_{forbidden} \tag{3}$$

Process Interaction. Changing values of the three navigation dimensions corresponds to a state transition within the navigation space. Since such state transitions are *user-driven*, we denote them as *process interactions*. In our framework, process interactions are represented by vectors. Changing the view from "logic-based" to "time-based" (cf. Fig. 2c) constitutes an example of such an interaction. A *one-dimensional* process interaction, in turn, is an activity transforming a given navigation state into another one by changing the value of exactly one navigation dimension. We assume that $g, s, v \in \mathbb{N}$ and $\mathbf{e} = (\tilde{e_1}, \tilde{e_2}, \tilde{e_3})$. A one-dimensional process navigation Int_{oneDim} can then be defined as follows:

$$Int_{oneDim} = \{(\tilde{e_1}, \tilde{e_2}, \tilde{e_3}) | \tilde{e_1}, \tilde{e_2}, \tilde{e_3} \in \{0, 1, -1\} \text{ and } \|\mathbf{e}\| = 1\} \tag{4}$$

In turn, a *multi-dimensional* process interaction can be defined as an activity transforming one navigation state into another by changing the value of several navigation dimensions at the same time (e.g., both the geographic dimension and the semantic dimension may be changed at once). Google Earth, for example, implicitly uses multi-dimensional interactions when the user applies the scroll wheel to zoom. If the geographic dimension is changed, the semantic dimension is changed accordingly. Since this functionality is well known and accepted by users, we apply it to process navigation as well. We define multi-dimensional process interaction as follows:

$$Int_{multiDim} = \{(\tilde{e_1}, \tilde{e_2}, \tilde{e_3}) | \tilde{e_1}, \tilde{e_2}, \tilde{e_3} \in \{0, 1, -1\}\} \tag{5}$$

Navigation Model (NM). In our framework, a navigation model corresponds to a pre-defined set of allowed process interactions. This set may contain one-dimensional as well as multi-dimensional process interactions. According to (4) and (5), and due to the subset relation between one- and multi-dimensional process interactions (6a), the set of all possible process interactions Int_{total} can be defined as follows:

$$Int_{oneDim} \subset Int_{multiDim} \tag{6a}$$

$$\Rightarrow Int_{total} = Int_{multiDim} \tag{6b}$$

This set of allowed process interactions can be further reduced by manually discarding the set of forbidden process interactions $Int_{forbidden}$. Thus, NM can be defined as follows:

$$NM = Int_{total} \backslash Int_{forbidden} \tag{7}$$

Navigation Sequence (NavSeq). A navigation sequence is a sequence of process interactions. It describes the path along which the user navigates from a start navigation state NS_0 to an end navigation state NS_n:

$$\begin{gathered} NavSeq = (a_1, \ldots, a_n, NS_0, NS_n) \\ \text{with } a_1, \ldots, a_n \in NM \wedge NS_0, NS_n \in BM \end{gathered} \tag{8}$$

Process Navigation (PN). Finally, process navigation can be defined as 4-tuple consisting of the basis model, the navigation model, a start state NS_0, and a navigation sequence defined by the user:

$$PN(BM, NM, NS_0, NavSeq) \tag{9}$$

Navigation sequence $NavSeq$ can be further investigated by applying linear algebra to the process navigation 4-tuple in (9) (cf. Section 3.1).

3 Process Navigation Framework

Basically, our process navigation framework comprises two main components (cf. Fig. 3): the *navigation* and *presentation layers*. The navigation layer specifies the basis model and the navigation model (cf. Section 2) using linear algebra (i.e., the formal approach we apply). In turn, the presentation layer deals with the visualization of business processes and related process information. It also provides different stencil sets enabling different process visualizations.

Distinguishing between navigation and presentation layer allows us to apply different visualizations in the context of the same navigation logic. In turn, this increases the flexibility of our framework as companies often prefer specific process visualizations [6]. Focus of this paper is on the *navigation layer*.

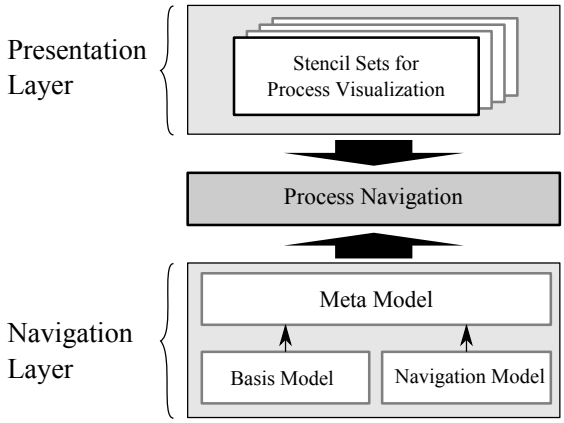

Fig. 3: Components of our process navigation framework.

3.1 Running Example and Basic Issues

We first present a running example—an automotive requirements engineering process (see Use Case 1 in Section 1). The navigation space, which is shown in Figure 4, has been manually designed. Its schematic model, which is based on the three navigation dimensions introduced in Section 2, is shown in the center of Fig. 4. It assumes that the requirements engineer is currently working on activity "Create Component Profile" within the process step "General Specification". Assume further that the requirement engineer needs to know the activity succeeding the current one in order to find the right contact person for passing the specification document resulting from his work. For this purpose, he navigates from his current context, i.e., the default start state $(0, 0, 0)$ to state $(1, 1, 0)$ in which he then can gather the information needed.

Concerning the three dimensions of this simple example, we can define $g, s, v \in \{0, 1\}$ instead of using natural numbers, i.e., every dimension is scaled in only two values. The overall number of possible navigation states is thus 2^3; note that in more complex navigation spaces, the number of navigation states increases exponentially with increasing number of navigation dimensions. In the following, NS_{total} is manually restricted by excluding two states: $(0, 1, 0)$ and $(0, 1, 1)$. These two states provide too many information items on the screen and would thus confuse the user. Think again of the Google Earth scenario, where all city names are shown in the semantic dimension, but the whole globe is shown in the geographic dimension at the same time. Considering (10a) and (10b), the basis model BM can be defined as shown in (10c):

$$NS_{total} = \{(0, 0, 0), (0, 0, 1), \ldots, (1, 1, 1)\} \tag{10a}$$

$$NS_{forbidden} = \{(0, 1, 0), (0, 1, 1)\} \tag{10b}$$

$$BM = \{(0, 0, 0), (0, 0, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\} \tag{10c}$$

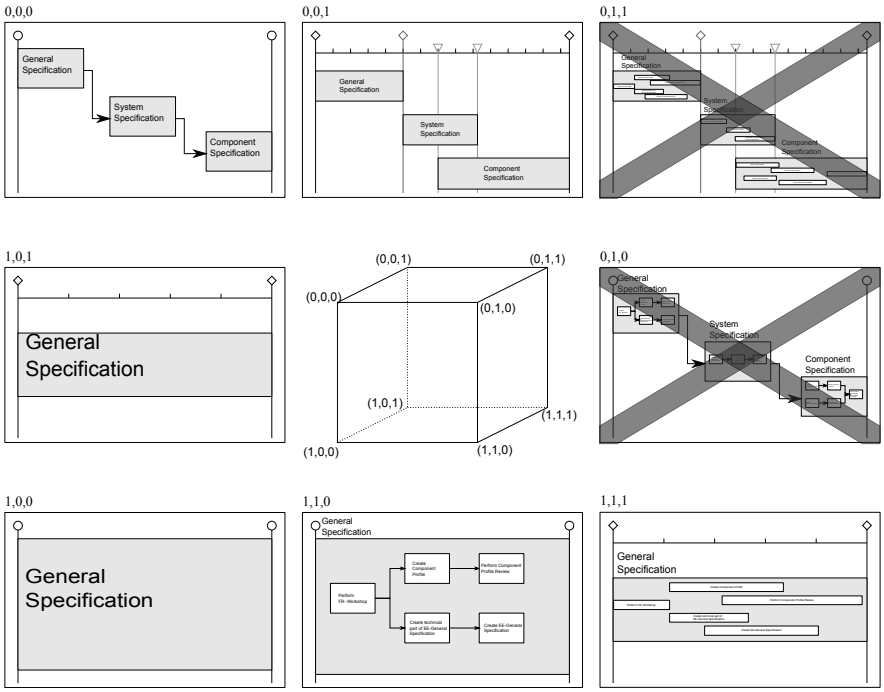

Fig. 4: Running example illustrating a navigation space with 8 navigation states.

In our running example, we only allow for one-dimensional process interactions. Therefore we restrict Int_{total} by excluding all other possible process interactions $Int_{forbidden}$:

$$NM = \left\{ a \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, a \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, a \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}, a \in \{1, -1\} \tag{11}$$

Based on the notation of process navigation (9), we can now investigate user-driven navigation sequences. For each process interaction, we can calculate whether or not the requirement engineer leaves the BM (i.e., reaches a navigation state not being an element of BM). We assume that he applies navigation sequence $NavSeq$:

$$NavSeq = \left(i_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, i_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right) \tag{12}$$

$NavSeq$ consists of two process interactions. More precisely, i_1 corresponds to a geographical zooming without changing the level of information detail, whereas i_2 corresponds to an increase of the level of information detail. In the following, we apply both navigation interactions to our BM.

Step 1: We first calculate navigation state NS_1 resulting after the first process interaction of the requirement engineer, i.e., after adjusting the geographic dimension to zoom into the process step "General Specification" (cf. Fig. 4). Therefore, we add the first vector i_1 to start state NS_0:

$$NS_0 + i_1 = NS_1 = (0,0,0)^T + (1,0,0)^T = (1,0,0)^T \tag{13}$$

As result, we obtain navigation state $(1,0,0) \in BM$. Hence, Step 1 constitutes a correct process interaction.

Step 2: From the newly obtained state NS_1 (i.e., the current start state) the requirement engineer now wants to increase the level of information detail, i.e., the value of the semantic dimension is increased to display the activities within the process step "General Specification". This process interaction i_2 is performed similar to Step 1:

$$NS_1 + i_2 = NS_2 = (1,0,0)^T + (0,1,0)^T = (1,1,0)^T \tag{14}$$

Finally, we check whether NS_2 is an element of BM. Since this is the case, $NavSeq$ can be constituted as allowed navigation sequence.

If the user chooses another navigation sequence to reach the preferred end state $(1,1,0)$, the result may be different. For example, a navigation sequence may start with increasing the value of the semantic dimension, i.e., by applying process interaction $(0,1,0)$. The resulting state will then be $(0,1,0)$, which is not an element of BM and thus constitutes a forbidden state, i.e., the state to which the user must not navigate. By calculating allowed navigation possibilities in advance, i.e., before the user action takes place, we can guide the user in not taking a forbidden way through the navigation space.

3.2 Navigation Possibilities

Taking our running example (cf. Fig. 4), we further investigate possibilities to navigate from a given navigation state to other states. This becomes necessary to effectively support users moving within the navigation space. Think of a scenario in which a user is initially situated in navigation state $(0,0,0)$. As navigation spaces could become more complex than in our running example, the user does not necessarily know how the basis model BM looks like in detail, i.e., he does not know to which navigation state(s) he may navigate. To avoid incorrect navigation, like the one from $(0,0,0)$ to forbidden state $(0,1,0)$, it is important to give recommendations regarding allowed navigation options in a given state. Considering navigation states, for example, it is important to identify neighboring navigation states allowed.

The *neighbor* characteristic describes the relation between two navigation states P_1 and P_2 that can be reached by applying exactly one single process interaction. Since we differentiate between one- and multi-dimensional process interactions, we also distinguish between one- and multi-dimensional neighbors.

One-dimensional neighbors. Two navigation states P_1 and P_2 constitute one-dimensional neighbors if a user can navigate from P_1 to P_2 by applying exactly one one-dimensional process interaction. In case only one-dimensional process interactions are allowed, the user may only navigate to one-dimensional neighbors of the current state:

$$\begin{gathered} P_1 \text{ is a one-dimensional neighbor of } P_2 \text{ iff} \\ P_1, P_2 \in BM \wedge \exists \mathbf{e} \in Int_{oneDim} : P_1 + \mathbf{e} = P_2 \end{gathered} \tag{15}$$

Multi-dimensional neighbors. Consider again our running example (cf. Fig. 4) and assume a user wants to navigate from $(0, 0, 0)$ to $(1, 1, 0)$. This could be accomplished by two consecutive one-dimensional process interactions. Generally, two states P_1 and P_2 are multi-dimensional neighbors, if P_2 is reachable from P_1 through a multi-dimensional process interaction:

$$\begin{gathered} P_1 \text{ is multi-dimensional neighbor of } P_2 \text{ iff} \\ P_1, P_2 \in BM \wedge \exists \mathbf{e} \in Int_{multiDim} : P_1 + \mathbf{e} = P_2 \end{gathered} \tag{16}$$

Reachable navigation states. A state P_2 is reachable from a state P_1 if there exists a navigation sequence that allows the user to navigate from P_1 to P_2. Thereby, the neighbor characteristics are applied in every process navigation step. As only pre-condition both P_1 and P_2 must be elements of BM:

$$\begin{gathered} P_1 \text{ is reachable from } P_2 \text{ iff} \\ P_1, P_2 \in BM \wedge \exists (n_1, \ldots, n_z) \text{ with } n_1, \ldots, n_z \in Int_{multiDim} \\ \wedge\ P_1 + \textstyle\sum_{i=1}^{z} n_i = P_2 \wedge P_1 + \sum_{i=1}^{m} n_i \in BM\ \forall m \end{gathered} \tag{17}$$

Knowing neighboring and reachable navigation states the navigation possibilities of a user can be determined. If a user is currently in a certain navigation state, we can guide further navigation interactions by recommending possible neighbors. This prevents users from a trial-and-error navigation.

3.3 Navigation Distance

Obviously, a navigation sequence applied by a user reflects the number of conducted process interactions. In turn, respective process interactions may require several user interactions (e.g., mouse clicks within an intranet portal). For the sake of simplicity assume that a user only applies one-dimensional process interactions. Then the number of user interactions corresponds to the number of mouse clicks. To decrease the latter (i.e., to enable a more efficient process navigation), the length of a navigation sequence required to navigate from a start state to a desired target state should be minimized. In the following, a method and metric to measure the length of navigation sequences are introduced.

As mentioned in Section 2, in general, we assume that the values of each navigation dimension correspond to any natural numbers. Hence, the distance between two arbitrary navigation states P_1 and P_2 can be easily calculated:

$$DIST(P_1, P_2) = \sqrt{(g_1 - g_2)^2 + (s_1 - s_2)^2 + (v_1 - v_2)^2} \quad \text{with } P_i = (g_i, s_i, v_i) \; ; \; i = 1, 2 \tag{18}$$

Note that this metric can be applied to arbitrary states of the navigation space, i.e., these two states do not necessarily have to be one- or multi-dimensional neighbors. Furthermore, we can measure the overall walking distance of a user navigating within the navigation space. This distance corresponds to the sum of one- or multi-dimensional process interactions:

$$NAVDIST(NavSeq) = \sum_{i=1}^{n} \|a_i\| \text{ where } a_1, \ldots, a_n \in NavSeq \tag{19}$$

3.4 Navigation Quality

To gain information about the quality of a chosen navigation sequence, we can measure its effectiveness , i.e., how quickly the user reaches his navigation goal when applying this navigation sequence. For this purpose, we consider the ratio of the distance between the start and end point of the navigation sequence on the one hand and its length on the other hand. Note that this metric does not only allow us to compare different navigation sequences, but it also enables better user assistence, e.g., based on recommendations about shorter navigation sequences:

$$Eff(P_1, P_2, NavSeq) = \frac{DIST(P_1, P_2)}{NAVDIST(NavSeq)} \tag{20}$$

3.5 Discussion

When assisting users in searching for process information in complex business processes, we are facing various challenges. One challenge is to categorize available process information. Introducing different navigation dimensions simplifies this task.

ID	Requirement
R1	The dynamic adoption of different navigation paths must be enabled.
R2	Shortcuts and favorites are needed.
R3	Fast and lean calculation of new navigation situations is needed.
R4	User actions must be easily traceable.
R5	An expansion of the navigation space (up to n dimensions) must be supported.

Table 1: Table of requirements.

Another challenge is to cope with the huge amount of process information and its classification as well as the formalization of process navigation. For the latter, several techniques may be applied. Table 1 lists main requirements on process

navigation, we previously identified in two case studies [8]. Table 2 further shows that linear algebra, unlike other potential formalisms we can use for formalizing our navigation approach, fulfills all five requirements. Linear algebra is both generic enough to support the future expansion of our navigation approach (R5) and lean enough to allow a fast adaption to new navigation situations (R3). Therefore, linear algebra is most suitable in our context.

ID	Finite state machines	Petri nets	State transition systems	Linear algebra	Predicate logic
R1	+++	+++	+++	+++	++
R2	++	++	++	+++	+
R3	++	++	+	+++	+
R4	+++	+++	+++	+++	++
R5	+	+	+	+++	+

Table 2: Comparison of different formalization techniques.
(+++: very good, ++: good, +:neutral)

4 Applying the Framework

We apply our navigation framework to a complex scenario comprising a large number of possible navigation states. Figure 5a shows a two-dimensional snippet of a three-dimensional navigation space. Black dots represent the basis model BM, i.e., the set of allowed navigation states. In turn, blank dots represent forbidden navigation states from set $NS_{forbidden}$. The horizontal dimension corresponds to the semantic and vertical one to the geographic dimension.

Assume that a user wants to navigate from start state $(0, 0, 0)$ to end state $(1, 8, 0)$. This corresponds to Use Case 2 from Section 1, where the project manager tries to identify project delays. Therefore, he needs detailed information about due dates, durations, and responsible persons (semantic dimension). Additionally, he requires an overview of all process steps of the project (geographic dimension).

First, we investigate the reachability of the end state from the start state. This way, we can check whether the needed information can be displayed at the desired geographic level, which shows all the process steps of the project.

When being in state $(0, 5, 0)$, a further increase of the information detail would result in an information overflow. The project manager then has to change the geographic level to zoom in, before he might be further allowed to increase the level of detail in the semantic dimension.

Second, we measure the distance between start and end state (cf. Fig. 5a):

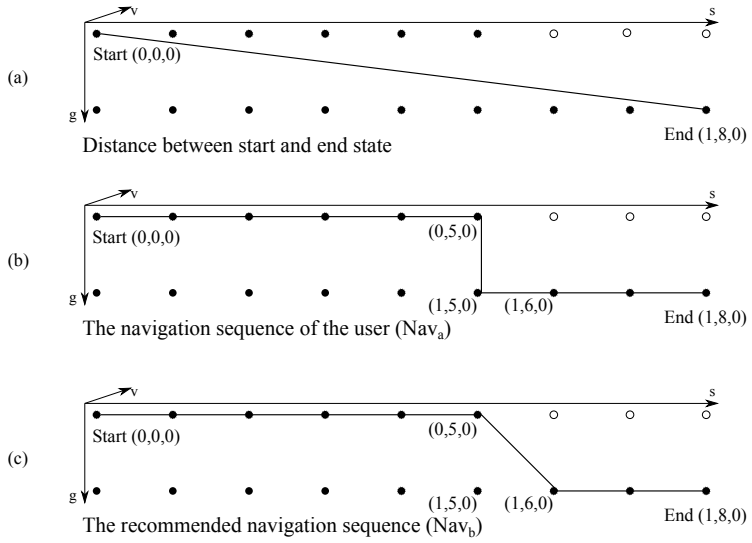

Fig. 5: Example of calculating the quality of navigation sequences.

$$DIST(Start, End) = \sqrt{8^2 + 1^2} \approx 8{,}06 \tag{21}$$

We now investigate the manager's navigation path while navigating within the navigation space, i.e., navigation sequence Nav_a from Fig. 5b. The manager applies nine one-dimensional process interactions to reach the end state. Hence, the distance is as follows:

$$DIST(Nav_a) = \sum_{i=0}^{n-1} DIST(P_{i+1}, P_i) = \sum_{0}^{8} 1 = 9 \tag{22}$$

Regarding our use case, the project manager might only be interested in adjusting the semantic dimension. The geographic dimension could be adjusted accordingly (from navigation state $(0, 5, 0)$ to state $(1, 6, 0)$) to avoid an overflow of the display with information. In this context, a multi-dimensional process interaction is applied automatically, reducing the user path by one interaction (cf. Fig. 5c). The distance of Nav_b can then be calculated as follows:

$$DIST(Nav_b) = \sum_{i=0}^{n-1} DIST(P_{i+1}, P_i) = 1{+}1{+}1{+}1{+}1{+}1{+}\sqrt{2}{+}1 \approx 8{,}41 \tag{23}$$

Using Eff, the following effectiveness ratios can be calculated for Nav_a and Nav_b respectively:

$$Eff(Start, End, Nav_a) = \frac{8{,}06}{9} \approx 89{,}55\% \tag{24a}$$

$$Eff(Start, End, Nav_b) = \frac{8{,}06}{8{,}41} \approx 95{,}79\% \tag{24b}$$

As can be seen in (24a) and (24b), suggesting navigation shortcuts leads to a more effective navigation path in Nav_b as indicated by the effectiveness ratios 95,79% and 89,55% respectively. This effect increases with the number of shortcuts. If typical navigation paths can be assigned to specific roles, further path suggestions could already be made before the user starts to navigate.

This example also indicates how our process navigation framework can be applied to Use Case 1. Again we use neighbors to measure distances and to calculate the effectiveness of navigation sequences. Doing so, more efficient navigation becomes possible by reducing unnecessary process interactions.

5 Related Work

According to Figure 6, related work stems from four areas:

First, *information retrieval* concerns information-seeking behavior of users [9]. We adopt this understanding in our process navigation framework. In this context, Belkin et al. [10] state that there are only little differences between information retrieval and information filtering. We apply this idea, by providing different navigation dimensions in our navigation framework.

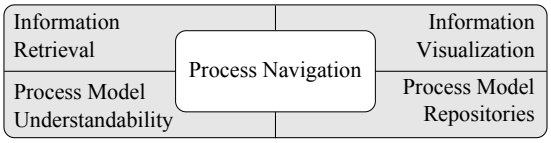

Fig. 6: Relevant areas of related work.

Second, *zoomable user interfaces* [11] have been developed to allow users to dynamically change views on information (*information visualization*). Specifically, they enable a decreasing fraction of an information space with an increasing magnification. Respective user interface concepts have been realized, for example, in *Squidy*, a zoomable design environment for natural user interfaces [12], in *ZEUS*, a zoomable explorative user interface for searching and presenting objects [13], and in *ZOIL*, a cross-platform user interface paradigm for personal information management [14]. Bederson also uses zooming techniques from JAZZ [15] and Pad++ [16] to develop intuitive user interfaces. Finally, Proviado applies aggregation and reduction techniques for creating views on large process models [17]. We adopt ideas from these approaches and extend them to ensure flexible process navigation.

Third, *process model repositories* [2, 18] are discussed in literature. Current repositories, however, suffer from redundancies and complexity making changes costly and error-prone [19]. Recently, several approaches have been suggested to improve this situation [20, 21].

Fourth, Mendling et al. [22] give insights into factors making process models better understandable. They investigate understandability as a proxy for the

quality of process models, e.g., a relatively high number of arcs has a negative effect on a process model's understandability.

6 Summary and Outlook

Quickly finding the process information needed during process execution is crucial for knowledge workers. To support them in accomplishing this task, companies crave for new ways of delivering and visualizing processes together with associated process information. This paper has presented a framework for navigating in large process spaces and related process information on different levels of detail. Our framework allows achieving flexible navigation goals for users with different roles and different tasks. Specifically, we use linear algebra to formalize our framework and apply it to selected use cases. Our results show, how our process navigation framework facilitates information retrieval in complex processes and related process information.

Future research will address three topics. First, we will specify the presentation layer (cf. Fig. 3) in more detail, e.g., by defining and developing sophisticated concepts for process-oriented information visualization. Second, we will develop concepts for integrating the navigation layer and the presentation layer. Third, we will focus on the evaluation of the process navigation approach by performing user tests and surveys.

References

1. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In: Proc. 4th Int'l Conf. on Business Process Management (BPM'06), LNCS 4102, Springer, pp. 368-377. (2006)
2. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Computers in Industry, 62(5), pp. 467-486. (2011)
3. Fauvet, M.C., Rosa, M.L., Sadegh, M., Alshareef, A., Dijkman, R.M., García-Bañuelos, L., Reijers, H.A., van der Aalst, W.M.P., Dumas, M., Mendling, J.: Managing process model collections with apromore. In: Proc. 8th Int'l Conf. on Service Oriented Computing (ICSOC'10), LNCS 6470, Springer, pp. 699–701. (2010)
4. van Wijk, J.J., Nuij, W.A.A.: Smooth and efficient zooming and panning. In: IEEE Symp. on Inf. Visualization (INFOVIS), IEEE Comp. Society, pp. 15–23. (2003)
5. Reichert, M., Bassil, S., Bobrik, R., Bauer, T.: The Proviado access control model for business process monitoring components. Enterprise Modelling and Information Systems Architectures (EMISA), 5(3), pp. 64–88. (2010)
6. Bobrik, R., Bauer, T., Reichert, M.: Proviado – personalized and configurable visualizations of business processes. In: Proc. 7th Int'l Conf. on Electronic Commerce and Web Technologies (EC-WEB'06), LNCS 4082, Springer, pp. 61–71. (2006)
7. Hipp, M., Mutschler, B., Reichert, M.: Navigating in process model collections: A new approach inspired by google earth. In: Proc. 1st Int'l Workshop on Process Model Collections (PMC'11), LNBIP 100, Springer, pp. 87–98. (2011)

8. Hipp, M., Mutschler, B., Reichert, M.: On the context-aware, personalized delivery of process information: Viewpoints, problems, and requirements. In: Proc. 6th Int'l Conf. on Availability, Reliability and Security (ARES'11), IEEE, pp. 390–397. (2011)
9. Belkin, N.J., Croft, W.B.: Information filtering and information retrieval: two sides of the same coin? Communications of the ACM (CACM), 35(12), pp. 29–38. (1992)
10. Belkin, N.J.: Interaction with texts: Information retrieval as information-seeking behavior. In: Proc. 1st Conf. on Information Retrieval '93, pp. 55–66. (1993)
11. Reiterer, H., Büring, T.: Zooming techniques. In: Encyclopedia of Database Systems, Springer, pp. 3684–3689. (2009)
12. König, W.A., Rädle, R., Reiterer, H.: Squidy: a zoomable design environment for natural user interfaces. In: Proc. 27th Int'l Conf. on Human Factors in Computing Systems (CHI'09), ACM, pp. 4561–4566. (2009)
13. Gundelsweiler, F., Memmel, T., Reiterer, H.: ZEUS - zoomable explorative user interface for searching and object presentation. In: Symposium on Human Interface (HCI(8)), LNCS 4557, Springer, pp. 288–297. (2007)
14. Zöllner, M., Jetter, H.C., Reiterer, H.: ZOIL: A design paradigm and software framework for post-WIMP distributed user interfaces. In: Distributed User Interfaces, HCI Series, Springer, pp. 87–94. (2011)
15. Bederson, B.B., Meyer, J., Good, L.: Jazz: an extensible zoomable user interface graphics toolkit in java. In: Proc. 13th ACM Symposium on User Interface Software and Technology (UIST), ACM, pp. 171–180. (2000)
16. Bederson, B.B., Hollan, J.D.: Pad++: A zooming graphical interface for exploring alternate interface physics. In: Proc. 7th ACM Symposium on User Interface Software and Technology (UIST), ACM, pp. 17–26. (1994)
17. Reichert, M., Kolb, J., Bobrik, R., Bauer, T.: Enabling personalized visualization of large business processes through parameterizable views. In: Proc. 27th ACM Symposium On Applied Computing (SAC'12), 9th Enterprise Engineering Track, ACM, pp. 1653–1660. (2012)
18. Rinderle-Ma, S., Kabicher, S., Ly, L.T.: Activity-oriented clustering techniques in large process and compliance rule repositories. In: Proc. 1st Int'l Workshop on Process Model Collections (PMC'11), LNBIP 100, Springer, pp. 14–25. (2011)
19. Weber, B., Reichert, M.: Refactoring process models in large process repositories. In: Proc. 20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08), LNCS 5074, Springer, pp. 124–139. (2008)
20. Choi, I., Kim, K., Jang, M.: An xml-based process repository and process query language for integrated process management. Knowledge and Process Management 14(4), pp. 303–316. (2007)
21. Reuter, C., Dadam, P., Rudolph, S., Deiters, W., Trillsch, S.: Guarded process spaces (gps): A navigation system towards creation and dynamic change of healthcare processes from the end-user's perspective. In: Proc. 4th Int'l Workshop on Process-Oriented Information Systems in Healthcare (ProHealth'11), LNBIP 100, Springer, pp. 237-248. (2011)
22. Mendling, J., Reijers, H.A., Cardoso, J.: What makes process models understandable? In: Proc. 5th Int'l Conf. on Business Process Management (BPM'07), LNCS 4714, Springer, pp. 48–63. (2007)